

Natural Language Processing

Lab 8: Naive Bayes and Text Classification

Objectives

- Understand the theoretical foundation of Naive Bayes classifiers for text classification.
- Implement a multinomial Naive Bayes classifier.
- Apply Naive Bayes to text classification problems.
- Evaluate classifier performance using precision, recall, and F1-score.
- Gain hands-on experience with text preprocessing, feature extraction, and model evaluation.

Naive Bayes Classification

Naive Bayes is a probabilistic classifier, meaning that for a document d , out of all classes $c \in C$ the classifier returns the class \hat{c} which has the maximum posterior probability given the document.

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d)$$

Bayes' Rule

The intuition of Bayesian classification is to use Bayes' rule to transform Bayes Classification equation into other probabilities that have some useful properties. Bayes' rule is presented in below it gives us a way to break down any conditional probability $P(x|y)$ into three other probabilities:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

- $P(c|d)$ = posterior probability of class c given document d
- $P(d|c)$ = likelihood of document d given class c
- $P(c)$ = prior probability of class c
- $P(d)$ = probability of document d (constant for all classes)

We can then substitute Bayes' Rule into Bayes Classification equation to get:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

We can conveniently simplify by dropping the denominator $P(d)$. This is possible because we will be computing $P(d|c)P(c)/P(d)$ for each possible class. But $P(d)$ doesn't change for each class; we are always asking about the most likely class for the same document d , which must have the same probability $P(d)$. Thus, we can choose the class that maximizes this simpler formula:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} P(d|c)P(c)$$

Bag of Words and Conditional Independence

For text classification, we represent documents as bags of words, ignoring word order. The Naive Bayes assumption states that features (words) are conditionally independent given the class:

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(f_1, f_2, \dots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

$$P(f_1, f_2, \dots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \dots \cdot P(f_n|c)$$

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f|c)$$

To apply the naive Bayes classifier to text, we need to consider word positions, by simply walking an index through every word position in the document:

positions \leftarrow all word positions in test document

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i | c)$$

Naive Bayes calculations, like calculations for language modeling, are done in log space, to avoid underflow and increase speed.

$$c_{NB} = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in \text{positions}} \log P(w_i | c)$$

Training

How can we learn the probabilities $P(c)$ and $P(w_i | c)$? Let's first consider the maximum likelihood estimate. We'll simply use the frequencies in the data. For the class prior $P(c)$ we ask what percentage of the documents in our training set are in each class c . Let N_c be the number of documents in our training data with class c and N_{doc} be the total number of documents. Then:

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

To learn the probability $P(w_i | c)$, we'll assume a feature is just the existence of a word in the document's bag of words, and so we'll want $P(w_i | c)$, which we compute as the fraction of times the word w_i appears among all words in all documents of topic c . We first concatenate all documents with category c into one big "category c " text. Then we use the frequency of w_i in this concatenated document to give a maximum likelihood estimate of the probability:

$$\hat{P}(w_i | c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

Here the vocabulary V consists of the union of all the word types in all classes, not just the words in one class c .

On Applying Laplace Smoothing:

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} = \frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V} \text{count}(w, c)) + |V|}$$

What do we do about words that occur in our test data but are not in our vocabulary at all because they did not occur in any training document in any class? The solution for such unknown words is to ignore them—remove them from the test unknown word document and not include any probability for them at all.

function TRAIN NAIVE BAYES(D, C) **returns** $\log P(c)$ and $\log P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms

N_{doc} = number of documents in D

N_c = number of documents from D in class c

$\text{logprior}[c] \leftarrow \log \frac{N_c}{N_{doc}}$

$V \leftarrow$ vocabulary of D

$\text{bigdoc}[c] \leftarrow$ **append**(d) **for** d \in D **with** class c

for each word w in V # Calculate $P(w|c)$ terms

$\text{count}(w, c) \leftarrow$ # of occurrences of w in $\text{bigdoc}[c]$

$\text{loglikelihood}[w, c] \leftarrow \log \frac{\text{count}(w, c) + 1}{\sum_{w' \in V} (\text{count}(w', c) + 1)}$

return logprior , loglikelihood , V

function TEST NAIVE BAYES(testdoc , logprior , loglikelihood , C, V) **returns** best c

for each class $c \in C$

$\text{sum}[c] \leftarrow \text{logprior}[c]$

for each position i in testdoc

$\text{word} \leftarrow \text{testdoc}[i]$

if word $\in V$

$\text{sum}[c] \leftarrow \text{sum}[c] + \text{loglikelihood}[\text{word}, c]$

return $\text{argmax}_c \text{sum}[c]$

Example

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

$$P(-) = \frac{3}{5} \quad P(+) = \frac{2}{5}$$

$$P(\text{"predictable"}|-) = \frac{1+1}{14+20} \quad P(\text{"predictable"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"no"}|-) = \frac{1+1}{14+20} \quad P(\text{"no"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"fun"}|-) = \frac{0+1}{14+20} \quad P(\text{"fun"}|+) = \frac{1+1}{9+20}$$

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Task 1

Implement a Naive Bayes Classifier model.

Evaluation Metrics

For binary classification, performance is measured using:

- **Precision:** Percentage of items detected that are actually positive

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **Recall:** Percentage of actual positive items correctly identified

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- **F-measure:** Harmonic mean of precision and recall

$$F_1 = \frac{2PR}{P+R}$$

Task 2

Evaluate your model implemented in Task 1.